**FRAGMENTATION**

Distributed Database systems provide distribution transparency of the data over the DBs. This is achieved by the concept called Data Fragmentation. That means, fragmenting the data over the network and over the DBs. Initially all the DBs and data are designed as per the standards of any database system – by applying normalization and denormalization. But the concept of distributed system makes these normalized data to be divided further. That means the main goal of DDBMS is to provide the data to the user from the nearest location to them and as fast as possible. Hence the data in a table are divided according their location or as per user’s requirement.

Dividing the whole table data into smaller chunks and storing them in different DBs in the DDBMS is called data fragmentation. By fragmenting the relation in DB allows:

* **Easy usage of Data:** It makes most frequently accessed set of data near to the user. Hence these data can be accessed easily as and when required by them.
* **Efficiency :** It in turn increases the efficiency of the query by reducing the size of the table to smaller subset and making them available with less network access time.
* **Security :** It provides security to the data. That means only valid and useful records will be available to the actual user. The DB near to the user will not have any unwanted data in their DB. It will contain only that information, which are necessary for them.
* **Parallelism :** Fragmentation allows user to access the same table at the same time from different locations. Users at different locations will be accessing the same table in the DB at their location, seeing the data that are meant for them. If they are accessing the table at one location, then they must wait for the locks to perform their transactions.
* **Reliability :** It increases the reliability of fetching the data. If the users are located at different locations accessing the single DB, then there will be huge network load. This will not guarantee that correct records are fetched and returned to the user. Accessing the fragment of data in the nearest DB will reduce the risk of data loss and correctness of data.
* **Balanced Storage :** Data will be distributed evenly among the databases in DDB.

Information about the fragmentation of the data is stored in DDC. When user sends a query, this DDC will determine which fragment to be accessed and it points that data fragment.

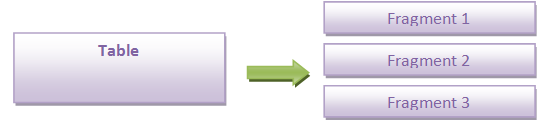
Fragmentation of data can be done according to the DBs and user requirement. But while fragmenting the data, below points should be kept in mind :

* **Completeness** **:** While creating the fragment, partial records in the table should not be considered. Fragmentation should be performed on whole table’s data to get the correct result. For example, if we are creating fragment on EMPLOYEE table, then we need to consider whole EMPLOYEE table for constructing fragments. It should not be created on the subset of EMPLOYEE records.
* **Reconstructions :** When all the fragments are combined, it should give whole table’s data. That means whole table should be able to reconstruct using all fragments. For example all fragments’ of EMPLOYEE table in the DB, when combined should give complete EMPLOYEE table records.
* **Disjointedness :** There should not be any overlapping data in the fragments. If so, it will be difficult to maintain the consistency of the data. Effort needs to be put to create same replication in all the copies of data. Suppose we have fragments on EMPLOYEE table based on location then, there should not be any two fragments having the details of same employee.

There are 3 types of data fragmentations in DDBMS.

* **Horizontal Data Fragmentation :**

As the name suggests, here the data / records are fragmented horizontally. i.e.; horizontal subset of table data is created and are stored in different database in DDB.



For example, consider the employees working at different locations of the organization like India, USA, UK etc. number of employees from all these locations are not a small number. They are huge in number. When any details of any one employee are required, whole table needs to be accessed to get the information. Again the employee table may present in any location in the world. But the concept of DDB is to place the data in the nearest DB so that it will be accessed quickly. Hence what we do is divide the entire employee table data horizontally based on the location. i.e.;

SELECT **\*** FROM EMPLOYEE WHERE EMP\_LOCATION = ‘INDIA;

SELECT **\*** FROM EMPLOYEE WHERE EMP\_LOCATION = ‘USA’;

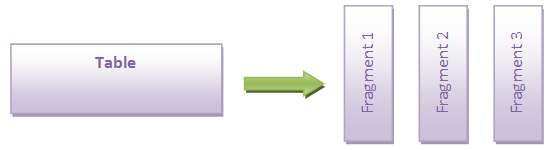
SELECT **\*** FROM EMPLOYEE WHERE EMP\_LOCATION = ‘UK;

Now these queries will give the subset of records from EMPLOYEE table depending on the location of the employees. These sub set of data will be stored in the DBs at respective locations. Any insert, update and delete on the employee records will be done on the DBs at their location and it will be synched with the main table at regular intervals.

Above is the simple example of horizontal fragmentation. This fragmentation can be done with more than one conditions joined by AND or OR clause. Fragmentation is done based on the requirement and the purpose of DDB.

* **Vertical Data Fragmentation :**

This is the vertical subset of a relation. That means a relation / table is fragmented by considering the columns of it.



For example consider the EMPLOYEE table with ID, Name, Address, Age, location, DeptID, ProjID. The vertical fragmentation of this table may be dividing the table into different tables with one or more columns from EMPLOYEE.

SELECT EMP\_ID, EMP \_FIRST\_NAME, EMP\_LAST\_NAME, AGE FROM EMPLOYEE;

SELECT EMP\_ID, STREETNUM, TOWN, STATE, COUNTRY, PIN FROM EMPLOYEE;

SELECT EMP\_ID, DEPTID FROM EMPLOYEE;

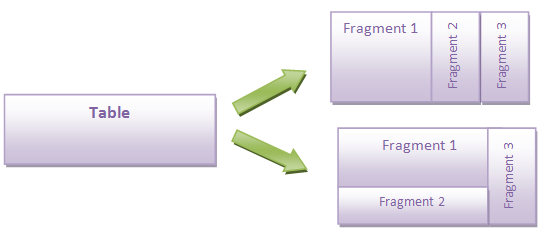
SELECT EMP\_ID, PROJID FROM EMPLOYEE;

This type of fragment will have fragmented details about whole employee. This will be useful when the user needs to query only few details about the employee. For example consider a query to find the department of the employee. This can be done by querying the third fragment of the table. Consider a query to find the name and age of an employee whose ID is given. This can be done by querying first fragment of the table. This will avoid performing ‘SELECT \*’ operation which will need lot of memory to query the whole table – to traverse whole data as well as to hold all the columns.

In this fragment overlapping columns can be seen but these columns are primary key and are hardly changed throughout the life cycle of the record. Hence maintaining cost of this overlapping column is very least. In addition this column is required if we need to reconstruct the table or to pull the data from two fragments. Hence it still meets the conditions of fragmentation.

* **Hybrid Data Fragmentation :**

This is the combination of horizontal as well as vertical fragmentation. This type of fragmentation will have horizontal fragmentation to have subset of data to be distributed over the DB, and vertical fragmentation to have subset of columns of the table.



As we observe in above diagram, this type of fragmentation can be done in any order. It does not have any particular order. It is solely based on the user requirement. But it should satisfy fragmentation conditions.

Consider the EMPLOYEE table with below fragmentations.

SELECT EMP\_ID, EMP \_FIRST\_NAME, EMP\_LAST\_NAME, AGE

FROM EMPLOYEE WHERE EMP\_LOCATION = ‘INDIA;

SELECT EMP\_ID, DEPTID FROM EMPLOYEE WHERE EMP\_LOCATION = ‘INDIA;

SELECT EMP\_ID, EMP \_FIRST\_NAME, EMP\_LAST\_NAME, AGE

FROM EMPLOYEE WHERE EMP\_LOCATION = ‘US;

SELECT EMP\_ID, PROJID FROM EMPLOYEE WHERE EMP\_LOCATION = ‘US;

This is a hybrid or mixed fragmentation of EMPLOYEE table.